(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 September 2003 (12.09.2003)

PCT

(10) International Publication Number
WO 03/075576 A2

(51) International Patent Classification[7]: H04N 7/24

(21) International Application Number: PCT/JP03/02463

(22) International Filing Date: 4 March 2003 (04.03.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/361,331 5 March 2002 (05.03.2002) US

(71) Applicant *(for all designated States except US)*: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.** [JP/JP]; 1006, Oaza Kadoma, Kadoma-shi, Osaka 571-8501 (JP).

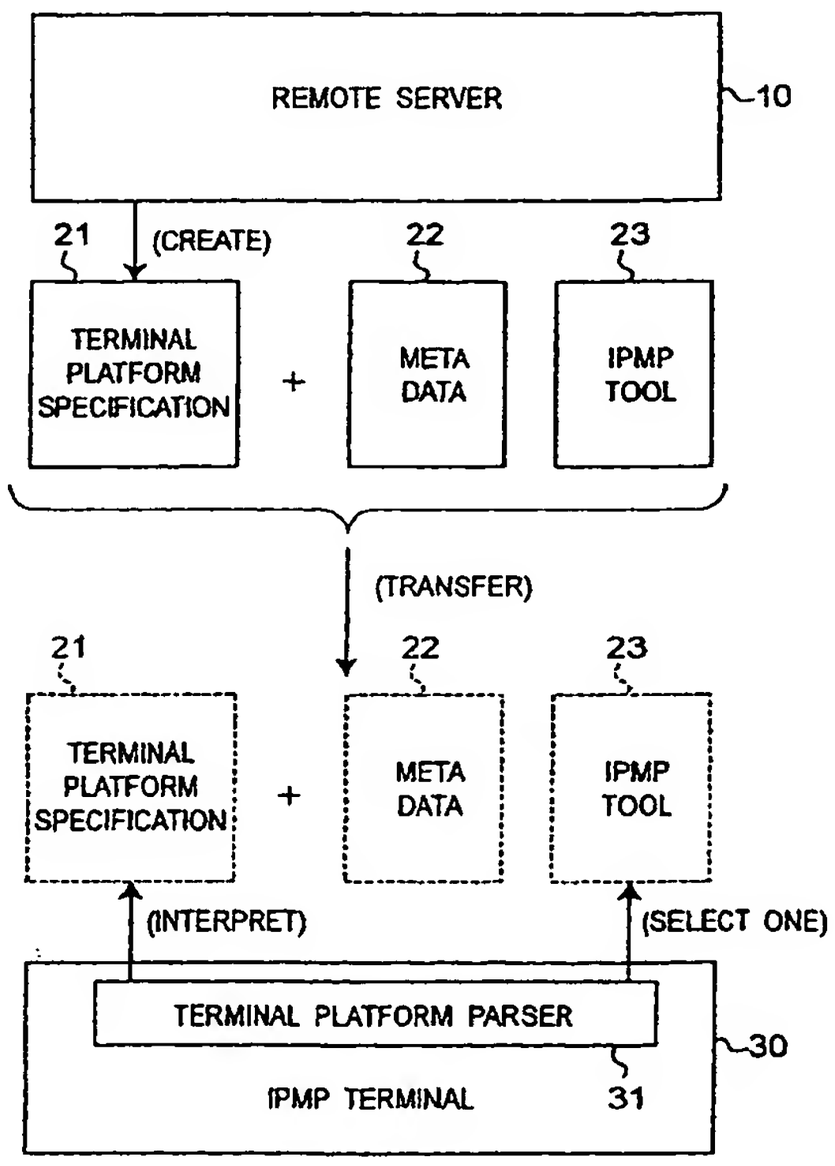
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **JI, Ming** [CN/SG]; Block 10, Geylang East Avenue 2, #02-09, 389758 Singapore (SG). **HUANG, Zhongyang** [CN/SG]; Block 608, Choa Chu Kang Street 62, #08-105, 680608 Singapore (SG). **SHEN, Sheng Mei** [SG/SG]; Block 20, Choa Chu Kang Street 64, #03-02 Windermere, 689093 Singapore (SG). **SENOH, Takanori** [JP/JP]; 1-24-8, Higashinakaburi, Hirakata-shi, Osaka 573-0093 (JP).

(74) Agents: **AOYAMA, Tamotsu** et al.; Aoyama & Partners, IMP Building, 3-7, Shiromi 1-chome, Chuo-ku, Osaka-shi, Osaka 540-0001 (JP).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

[Continued on next page]

(54) Title: A METHOD OF TRANSFERRING INFORMATION SPECIFYING A TOOL UTILIZED FOR PROCESSING A CONTENT PROTECTED BY IPMP



(57) Abstract: A method of transferring information specifying an IPMP tool utilized for processing a content protected by IPMP to an IPMP terminal specifies and uses XML schema for terminal platform specification, parametric description, and parametric aggregation which is to be used in MPEG-n IPMP system. For example, according to the method, based on a predetermined terminal platform XML schema and language to describe a terminal platform, terminal platform specification is created based on the terminal platform schema and language to specify on what platform or terminal the IPMP tool can run. The terminal platform specification is attached to metadata associated with the tool. The terminal platform specification and the IPMP tool are transferred to an IPMP terminal which is implemented with a terminal platform parser based on the terminal platform schema and language.

WO 03/075576 A2

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DESCRIPTION

A METHOD OF TRANSFERRING INFORMATION SPECIFYING A TOOL UTILIZED FOR PROCESSING A CONTENT PROTECTED BY IPMP

Technical Field

The present invention relates to content distribution and protection, especially to such applications where the protected content based on MPEG-n is consumed by different IPMP (Intellectual Property Management and Protection) terminals, and the same content is protected by different IPMP tools.

Background Art

Content distribution is becoming more and more demanding as multimedia data and contents can reach anywhere and anytime. Users are happy with the convenience and flexibility, and they can enjoy entertainment easily and efficiently. On the other hand, content owners are worried about the illegal usage of their property. There is a balance between two sides.

There are a lot of protection techniques for protecting the content, such as data encryption, watermarking, etc. They have been implemented in many content distribution applications. It seems different system employs different kinds of mechanisms and protection techniques to distribute content with protection. All the terminals or content consuming devices in that case are only able to play and consume the content that is provided by the same content provider. They cannot exchange their terminal or device to playback different contents.

In MPEG-n context, a standardisation group has been working on MPEG-2,4,7,21 IPMP (Intellectual Property Management and Protection).

Fig. 7 shows the prior art for the current typical IPMP architecture. In Fig. 7, it is assumed that content in module 1.1 coming into IPMP system is an MPEG-21 protected multimedia. In fact, under MPEG-21 "big picture", the coming content is called Digital Item in MPEG-21 framework. Three units 1.2, 1.3, 1.4 in module 1.1

illustrate IPMP information transferred in content for IPMP system processing. The IPMP Tool List unit 1.2 identifies, and enables selection of, the IPMP Tools required to process the content. The IPMP Tool Elementary Stream unit 1.3 identifies the actual tools carried in the content itself. IPMP Information unit 1.4 identifies the information directed to a given IPMP Tool to enable, assist or facilitate its operation.

As for the right part of Fig. 7, it shows an IPMP terminal (module 1.5, a device that consumes possibly protected incoming content in compliance with the usage rules) including IPMP Tool Manager (module 1.6, processes IPMP Tool List(s) unit 1.2 and retrieves the IPMP Tools module 1.8, 1.9, 1.10, 1.11 that are specified therein) and Message Router (module 1.7, implements the Terminal-side behavior of the Terminal-Tool interface). The module 1.8 of Missing IPMP Tools (not available in module 1.5) and the module 1.9, 1.10, 1.11 of IPMP Tool (available in module 1.5) denote the tool that perform (one or more) IPMP functions such as authentication, decryption, watermarking, etc.

Such a solution is able to achieve both of the following:

1. Allow the same protected content to be consumed on different vendors' MPEG-n IPMP Terminals. This will be fully enabled.

2. Allow the same content to be protected by different vendors' IPMP Tools. This will be assisted to as large extent as possible.

To achieve such a wide interoperability, IPMP provides download ability of tools, where tools can be retrieved remotely. IPMP also allows the terminal to choose its own favourite tool according to parametric description. An IPMP terminal can also aggregate several tools together to form a tool set (act as just one tool) according to some parametric aggregation.

An IPMP tool may also optionally carry a tool meta data that specifies information about this tool, for example, whether this tool can run on a particular operating system.

In order to achieve a flexible and interoperable IPMP system structure for MPEG-n and other applications as described above, we should be able to allow the followings.

1. When an IPMP terminal goes to a remote site trying to retrieve a missing IPMP tool in order to playback a certain content, the IPMP terminal should provide the remote site terminal platform specification, so that the remote site can send a tool that can run on this particular IPMP terminal.

2. An IPMP Tool's metadata may optionally include the terminal platform specification that specifies what are the terminals and platforms that the tool can run on.

3. A parametric description to allow terminal to choose a certain IPMP tool that can facilitate the job it is supposed to do.

4. A parametric aggregation to allow several tools to form as one tool set, and appeared as just one tool to the IPMP terminal.

Disclosure of Invention

To solve the problem, to assure a clear and interoperable standard, and to provide future extensibility, XML schema should be defined for terminal platform specification, parametric description as well as parametric aggregation.

The XML schema for terminal platform specification should allow specification of CPU, memory, Operating system, virtual machine, etc, so that the remote site can have a full knowledge of the terminal and choose an appropriate tool for the terminal.

When the tool metadata contains the terminal platform XML language, the terminal can also decide whether this tool can run on it according to the XML language.

The parametric description is defined to allow a generic description of any types of IPMP tool, whether it is a decryption tool or a watermarking tool or a rights management tool. Based on the parametric description, an IPMP terminal is then able to choose its own favourite tool.

The parametric aggregation is defined to allow the terminal (or master tool) to combine a set of tools, connection points, i/o, etc can all be defined in the parametric aggregation so that the terminal (or master tool) can have a full knowledge of what to do.

Based on the terminal platform XML schema, an IPMP terminal can describe what it is, what the CPU is, what the operating system is, whether there are any assistant hardwares, etc. The IPMP terminal can then send the information to the remote site while trying to retrieve a missing IPMP tool. The remote site, after receiving the information, and XML schema parsing, can then decide which tool (a windows DLL or a Java Byte Code, for example) to send.

Based on the terminal platform XML schema, an IPMP tool can also describe on what platform it can run, and carry the information in the metadata associated with the IPMP tool. When receiving such an IPMP tool, an IPMP terminal can decide whether this tool can surely run on it.

Based on the parametric description XML schema, the content provider can now describe what type of tool is required to playback the content, instead of using fixed tool IDs. For example, the content provider can specify that an AES tool, with block size of 128 bits is required to decrypt video stream. The IPMP terminal, upon receiving such XML language specifying this tool, can then choose an optimised AES tool from the embedded tools.

Based on the parametric aggregation XML schema, the content provider can describe what are the tools need to be combined as one tool to process the content. The XML language can specify things like tool IDs, tool i/o, etc. So that the terminal can use the information to build such a tool set.

Brief Description of Drawings

Fig. 1 is a view explaining the method in the fist embodiment of the present invention.

Fig. 2 is a view explaining the method in the second embodiment of the present invention.

Fig. 3 is a view explaining the method in the third embodiment of the present invention.

Fig. 4 is a view explaining the method in the fourth embodiment of the present invention.

Fig. 5 shows the architecture of Terminal Platform XML Schema.

Fig. 6 shows the architecture of Parametric Aggregation.

Fig. 7 shows IPMP architecture for generic IPMP as the prior art.

Best Mode for Carrying Out the Invention

The IPMP system has been designed to provide a framework and specifications for better interoperability in standard bodies, especially in MPEG-n (n=2, 4, 7, 21). It allows the same protected content to be consumed on different vendors' terminals; it also allows the same content to be protected by different vendors' IPMP Tools.

1. First Embodiment

With reference to Fig. 1, description is made below to one embodiment of a method of transferring information specifying a tool utilized for processing a content protected by IPMP from a server to an IPMP terminal. The method uses XML schema for terminal platform specification in MPEG-n IPMP system.

A flexible terminal platform schema and language are specified and defined to describe a terminal platform specification in advance. That is, the terminal platform schema and language are standardised to make them known to public. The terminal platform specification includes information describing, in XML schema, terminal functions, terminal performance and so on required for decrypting the content to which the specification is attached, and is described later.

A remote server 10 creates terminal platform specification 21 based on the predetermined terminal platform schema and language to specify on what platform or terminal the IPMP tool can run.

The created terminal platform specification 21 is attached to metadata 22 associated with the tool 23. The tool 23 together with the terminal platform specification 21 attached to the metadata 22 may be carried in an IPMP protected content.

The server 10 transfers the metadata 22 with the terminal platform specification 21 and the IPMP tool 23 to an IPMP terminal 30 which is implemented with a terminal platform parser 31 based on the terminal platform schema and language. The terminal platform parser 31 can interpret and execute the terminal platform specification 21 according to syntax of given XML schema.

After receiving the metadata 22 with the terminal platform specification 21, the IPMP terminal 30 extracts the terminal platform specification 21 from the metadata 22, and interprets the terminal platform specification 21 to determine whether the IPMP tool is the terminal's desired one, by the implemented terminal platform parser 31.

2. Second Embodiment

With reference to Fig. 2, description is made below to another embodiment of a method of transferring information specifying a tool utilized for processing a content protected by IPMP from a server to an IPMP terminal. The method uses XML schema for terminal platform specification in MPEG-n IPMP system.

An IPMP terminal 30 creates terminal platform specification 21 based on the predetermined terminal platform schema and language to specify itself. The IPMP terminal 30 then transfers the terminal platform specification 21 to a remote server 10 when the IPMP terminal 30 tries to retrieve an IPMP tool 23 from the remote server 10 which is implemented with a terminal platform parser 11 based on the terminal platform schema and language.

The server 10 receives the terminal platform specification 21 from the terminal 30, interprets the terminal platform specification 21, determines which IPMP tool should be sent back to the requesting IPMP terminal 30, and then sends the determined IPMP tool 23 to the requesting IPMP terminal 30. The terminal 30 receives the IPMP tool 23 and uses it to decrypt the content.

3. Third Embodiment

With reference to Fig. 3, description is made to still another embodiment of a method according to the invention. The method uses XML schema for parametric description in MPEG-n IPMP system. The parametric description is information describing, in XML schema, terminal functions, terminal performance and so on required for decrypting the content to which the specification is attached.

A flexible parametric description schema and language are specified and defined to describe an IPMP tool in advance. That is, the parametric description schema and language are standardised to make known to public.

A remote server 10 creates a parametric description 25 based on the predetermined parametric description schema and language to specify a desired IPMP tool, attaches the parametric description 25 to a content 26 that is protected by IPMP, and transfers the parametric description 25 and the content 26 to an IPMP terminal 30 which is implemented with a parametric description parser 32 based on the parametric description schema and language. The parametric description parser 32 can interpret and execute the parametric description according to syntax of given XML schema.

The terminal 30 extracts the parametric description 25 from the content 26, interprets the parametric description 25, and selects an IPMP tool from the terminal's tool set 35 that satisfy the parametric description, by the parametric description parser 32.

More specifically, when receiving the content, the parametric description parser 32 extracts the parametric description 25 from the received content 26, and routes the extracted parametric description 25 to each IPMP tool within its tool set 35 to ask whether the tool satisfy the parametric description 25. The parametric description parser 32 compares each IPMP tool within the terminal's tool set 35 with the parametric description 25 and verifies whether the IPMP tool can satisfy the parametric description 25. The result is made known to the IPMP terminal 30. The IPMP terminal 30 can then choose a tool from the tool set that satisfy the parametric description 25.

4. Fourth Embodiment

With reference to Fig. 4, description is made below to still another embodiment of a method according to the invention. The method uses XML schema for parametric aggregation in MPEG-n IPMP system. The parametric aggregation is described later with reference to Fig. 6.

A flexible parametric aggregation schema and language are specified and defined to describe a combination of IPMP tools in advance. That is, the parametric aggregation schema and language are standardised to make known to public.

A remote server 10 creates a parametric aggregation 27 based on the predetermined parametric aggregation schema and language to specify a desired combination of IPMP tools, and attaches the parametric aggregation 27 to a content 26 that is protected by IPMP. The server 10 transfers the parametric aggregation 27 and the content 26 to an IPMP terminal 30 which is implemented with a parametric aggregation parser 33 based on the parametric aggregation schema and language.

The IPMP terminal 30 extracts the parametric aggregation 27 from the content 26, interprets the parametric aggregation 27, and combines the specified IPMP tools from the terminal's tool set that satisfy the parametric aggregation 27, by the parametric aggregation parser 33.

5. Terminal Platform XML Schema

Fig 5 illustrates the architecture of the terminal platform in which a terminal ID is described in the following XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="TerminalID">
        <xsd:annotation>
            <xsd:documentation>Identification of a terminal</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
```

```
<xsd:element name="TerminalType">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Vendor" type="xsd:string"/>
            <xsd:element name="Model" type="xsd:string"/>
            <xsd:element name="SerialNO" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="OperatingSystem">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Vendor" type="xsd:string"/>
            <xsd:element name="Model" type="xsd:string"/>
            <xsd:element name="Version" type="xsd:string"/>
            <xsd:element name="SerialNO" type="xsd:string" minOccurs="0"/>
            <xsd:element name="VirtualMachine" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Vendor" type="xsd:string"/>
                        <xsd:element name="Name" type="xsd:string"/>
                        <xsd:element name="Version" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="CPU">
    <xsd:complexType>
```

```
<xsd:sequence>
    <xsd:element name="Vendor" type="xsd:string"/>
    <xsd:element name="Model" type="xsd:string"/>
    <xsd:element name="Speed" type="xsd:integer"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="Memory">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="Vendor" type="xsd:string"/>
    <xsd:element name="Model" type="xsd:string"/>
    <xsd:element name="Size" type="xsd:integer"/>
    <xsd:element name="Speed" type="xsd:integer"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="AsstHardware" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
    <xsd:element name="SmartCard" minOccurs="0">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Vendor" type="xsd:string"/>
                <xsd:element name="Model" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="HardKey" minOccurs="0">
        <xsd:complexType>
```

```
                    <xsd:sequence>
                        <xsd:element name="Type" type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="Network" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Type" type="xsd:string"/>
            <xsd:element name="Details" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<xsd:element name="RPCMechanism" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Type" type="xsd:string"/>
            <xsd:element name="Details" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="Type" type="xsd:string" use="required"/>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

The semantics of the description is as follows.

TerminalType

Vendor : "Vendor of the specific terminal, for example, a PC's vendor may be DELL"

Model : "Model name"

SerialNO : "serial number of this terminal"

OperatingSystem

Vendor : "Vendor of the specific operating system, for example, windows95's vendor is Microsoft"

Model : "model name of the operating system"

Version : "version number of the operating system"

SerialNO : "serial number of the operating system , if any"

VirtualMachine

Vendor : "Virtual Machine's vendor, for example, JVM's vendor is SUN"

Name : "name of the virtual machine, for example, Java Virtual Machine"

Version : "version number of the virtual machine"

CPU

Vendor : "Vendor of the CPU, for example, INTEL"

Model : "Model name of the CPU, for example, Pentium"

Speed : "Speed of the CPU"

Memory

Vendor : "Vendor of the memory"

Model : "Model name of the memory"

Size : "size of the memory"

Speed : "speed of the memory"

AsstHardware

SmartCard

Vendor : "Vendor of the smartcard"

Model : "Model Name of the smartcard"

HardKey

Type : "type of the hardkey, for example, parallel port hardkey"

Network

Type : "type of the network, for example, Ethernet"

Detail

RPCMechanism

Type : "mechanism of the Remote Procedure Call, for example, SOAP, DCOM"

Details

6. Parametric Description

The parametric description of the terminal platform is described in the following XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="PARAMETRIC_DESCRIPTION">
        <xsd:annotation>
            <xsd:documentation>Comment describing your root element</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="VERSION">
                    <xsd:complexType>
                        <xsd:attribute name="MAJOR_VERSION" type="xsd:string"/>
                        <xsd:attribute name="MINOR_VERSION" type="xsd:string"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="CLASS" type="xsd:string"/>
```

```
<xsd:element name="SUBCLASS" type="xsd:string"/>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="TYPEDATA" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:anySimpleType">
                            <xsd:attribute name="type" type="xsd:anySimpleType"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="type" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="ADDITIONAL_DATA" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

The semantics of the above description is as follows.

VERSION

MAJOR_VERSION : "Major version of the parametric description schema"

MINOR_VERSION : "minor version of the parametric description schema"

CLASS : "class of the parametrically described tool, for example, decryption"

SUBCLASS : "sub-class of the parametrically described tool, for example, AES under decryption class"

TYPEDATA : "specific type data to describe a particular type of tool, for example, Block_length of further specify a AES decryption tool"

type (attribute) : "value of the type data above, for example, 128 for the Block_length"

ADDITIONAL_DATA : "Any additional data which may help to further describe the parametrically defined tool"

7. Parametric Aggregation

Fig. 6 illustrates the architecture of the parametric aggregation which is described in the following XML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xsd:element name="PARAMETRIC_AGGREGATION">
        <xsd:annotation>
            <xsd:documentation>Comment describing your root element</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="VERSION">
                    <xsd:complexType>
                        <xsd:attribute name="MAJOR_VERSION" type="xsd:string"/>
                        <xsd:attribute name="MINOR_VERSION" type="xsd:string"/>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="CODE_LIST_IN">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Code" maxOccurs="unbounded">
```

```
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:anySimpleType">
                                <xsd:attribute name="code" type="xsd:string" use="required"/>
                                <xsd:attribute name="Interface_ID" type="xsd:integer" use="optional"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="CODE_LIST_OUT">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Code" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:simpleContent>
                                    <xsd:extension base="xsd:anySimpleType">
                                        <xsd:attribute name="code" type="xsd:string" use="required"/>
                                        <xsd:attribute name="Interface_ID" type="xsd:integer" use="optional"/>
                                    </xsd:extension>
                                </xsd:simpleContent>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="AGGREGATION_UNIT" maxOccurs="unbounded">
                <xsd:complexType>
```

```
<xsd:sequence>
    <xsd:element name="TOOL_ID" type="xsd:string"/>
    <xsd:element name="CODE_LIST_IN">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Code" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:anySimpleType">
                                <xsd:attribute name="code" type="xsd:string"
use="required"/>
                                <xsd:attribute name="Interface_ID" type="xsd:integer"
use="optional"/>
                            </xsd:extension>
                        </xsd:simpleContent>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="CODE_LIST_OUT">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Code" maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:simpleContent>
                            <xsd:extension base="xsd:anySimpleType">
                                <xsd:attribute name="code" type="xsd:string"
use="required"/>
```

```
<xsd:attribute name="Interface_ID" type="xsd:integer"
use="optional"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
    </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ADDITIONAL_DATA" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

8. Effect of Invention

This invention solves the following problems by specifying XML schemas for terminal platform specification, parametric description and parametric aggregation.

1. When an IPMP terminal goes to a remote site trying to retrieve a missing IPMP tool in order to playback a certain content, the IPMP terminal can use the terminal platform XML schema to provide the remote site terminal platform specification, so that the remote site can send a tool that can run on this particular IPMP terminal.

2. An IPMP Tool's metadata may optionally include the terminal platform specification using the terminal platform XML schema that specifies what are the terminals and platforms that the tool can run on.

3. A parametric description using the parametric description schema to allow terminal to choose a certain IPMP tool that can facilitate the job it is supposed to do.

4. A parametric aggregation using the parametric aggregation schema to allow several tools to form as one tool set, and appeared as just one tool to the IPMP terminal.

Although the present invention has been described in connection with specified embodiments thereof, many other modifications, corrections and applications are apparent to those skilled in the art. Therefore, the present invention is not limited by the disclosure provided herein but limited only to the scope of the appended claims.

CLAIMS

1. A method of transferring information specifying an IPMP tool utilized for processing a content protected by IPMP to an IPMP terminal, wherein the method uses XML schema for terminal platform specification in IPMP system, and comprises:

based on a predetermined terminal platform schema and language to describe a terminal platform,

creating terminal platform specification based on the terminal platform schema and language to specify on what platform or terminal the IPMP tool can run;

attaching the terminal platform specification to metadata associated with the tool; and

transferring the terminal platform specification and the IPMP tool to an IPMP terminal implemented with a terminal platform parser based on the terminal platform schema and language.

2. The method according to claim 1, further comprising, before the transferring, carrying the tool together with the terminal platform specification in the content protected in IPMP.

3. A method of transferring information specifying a tool utilized for processing a content protected by IPMP to an IPMP terminal, wherein the method uses XML schema for terminal platform specification in MPEG-n IPMP system,

based on a predetermined terminal platform schema and language to describe a terminal platform, the terminal platform specification is created at an IPMP terminal according to the terminal platform schema and language to specify the IPMP terminal, and the method comprises:

receiving from the IPMP terminal the terminal platform specification when the IPMP terminal tries to retrieve an IPMP tool from a remote server;

using a terminal platform parser based on the terminal platform schema and language; and

interpreting the terminal platform specification and determining which IPMP tool should be sent back to the requesting IPMP terminal.

4. Method of transferring information specifying a tool utilized for processing a content protected by IPMP to an IPMP terminal, wherein the method uses XML schema for parametric description in MPEG-n IPMP system, and comprises:

based on a predetermined parametric description schema and language to describe an IPMP tool;

creating a parametric description based on the parametric description schema and language to specify a desired IPMP tool;

attaching the parametric description to a content that is protected by IPMP; and

transferring the description and the content to an IPMP terminal implemented with a parametric description parser based on the parametric description schema and language.

5. Method of determining a tool to process a content protected by IPMP at an IPMP terminal, by using XML schema for parametric description in IPMP system, wherein

based on a predetermined parametric description schema and language to describe an IPMP tool, a parametric description is created according to the parametric description schema and language to specify a desired IPMP tool, and the parametric description is attached to the content that is protected by IPMP, and

the method comprises:

extracting the parametric description from the content;

routing the parametric description to each IPMP tool within the terminal's tool set to ask whether the tool satisfies the parametric description;

comparing the tool within the terminal's tool set with the parametric description to verify whether each IPMP tool can satisfy the parametric description; and

choosing a tool from the tool set that satisfy the parametric description.

6. Method of transferring information specifying a tool utilized for processing a content protected by IPMP to an IPMP terminal, wherein the method uses XML schema for parametric aggregation in MPEG-n IPMP system, and comprises:

based on a predetermined parametric aggregation schema and language to describe a combination of IPMP tools,

creating a parametric aggregation based on the parametric aggregation schema and language to specify a desired combination of IPMP tools.

attaching the parametric aggregation to content that is protected by IPMP to an IPMP terminal; and

transferring the aggregation and the content to an IPMP terminal implemented with a parametric aggregation parser based on the parametric aggregation schema and language.

Fig.1

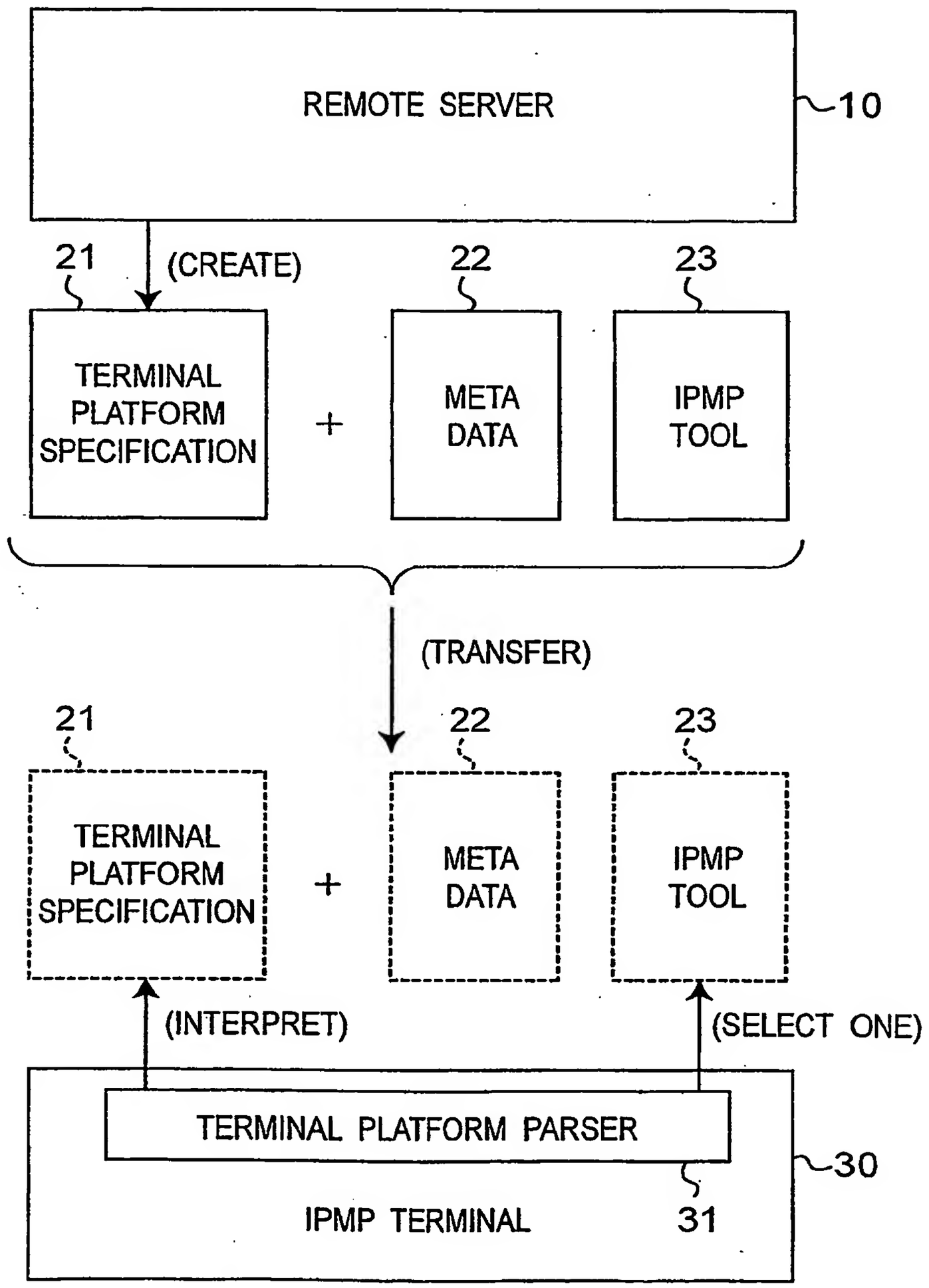

REMOTE SERVER
10
21
(CREATE)
22
23
TERMINAL PLATFORM SPECIFICATION
+
META DATA
IPMP TOOL
(TRANSFER)
21
22
23
TERMINAL PLATFORM SPECIFICATION
+
META DATA
IPMP TOOL
(INTERPRET)
(SELECT ONE)
TERMINAL PLATFORM PARSER
30
IPMP TERMINAL
31

Fig.2

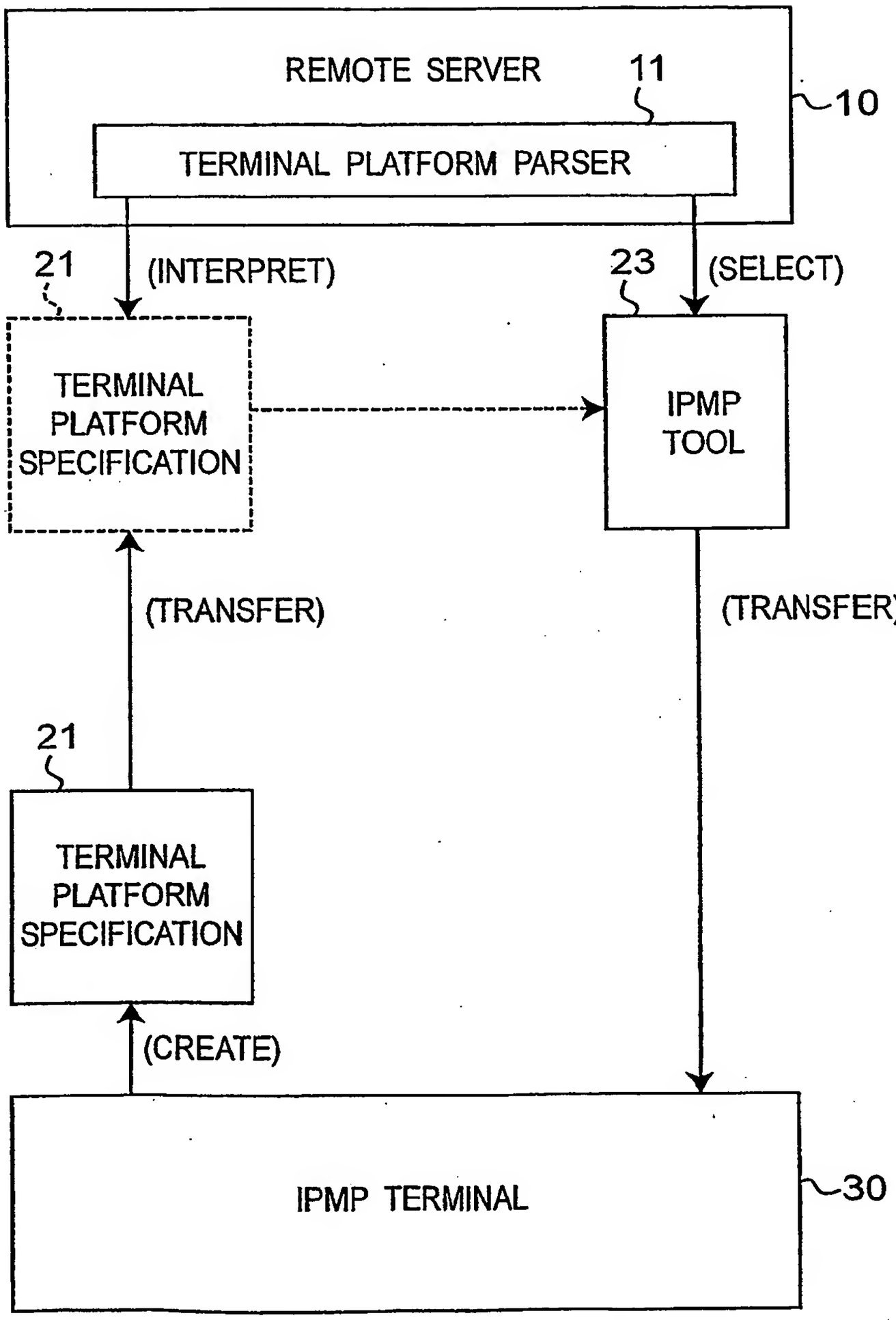

REMOTE SERVER
11
10
TERMINAL PLATFORM PARSER
21
(INTERPRET)
23
(SELECT)
TERMINAL PLATFORM SPECIFICATION
IPMP TOOL
(TRANSFER)
(TRANSFER)
21
TERMINAL PLATFORM SPECIFICATION
(CREATE)
IPMP TERMINAL
30

Fig.3

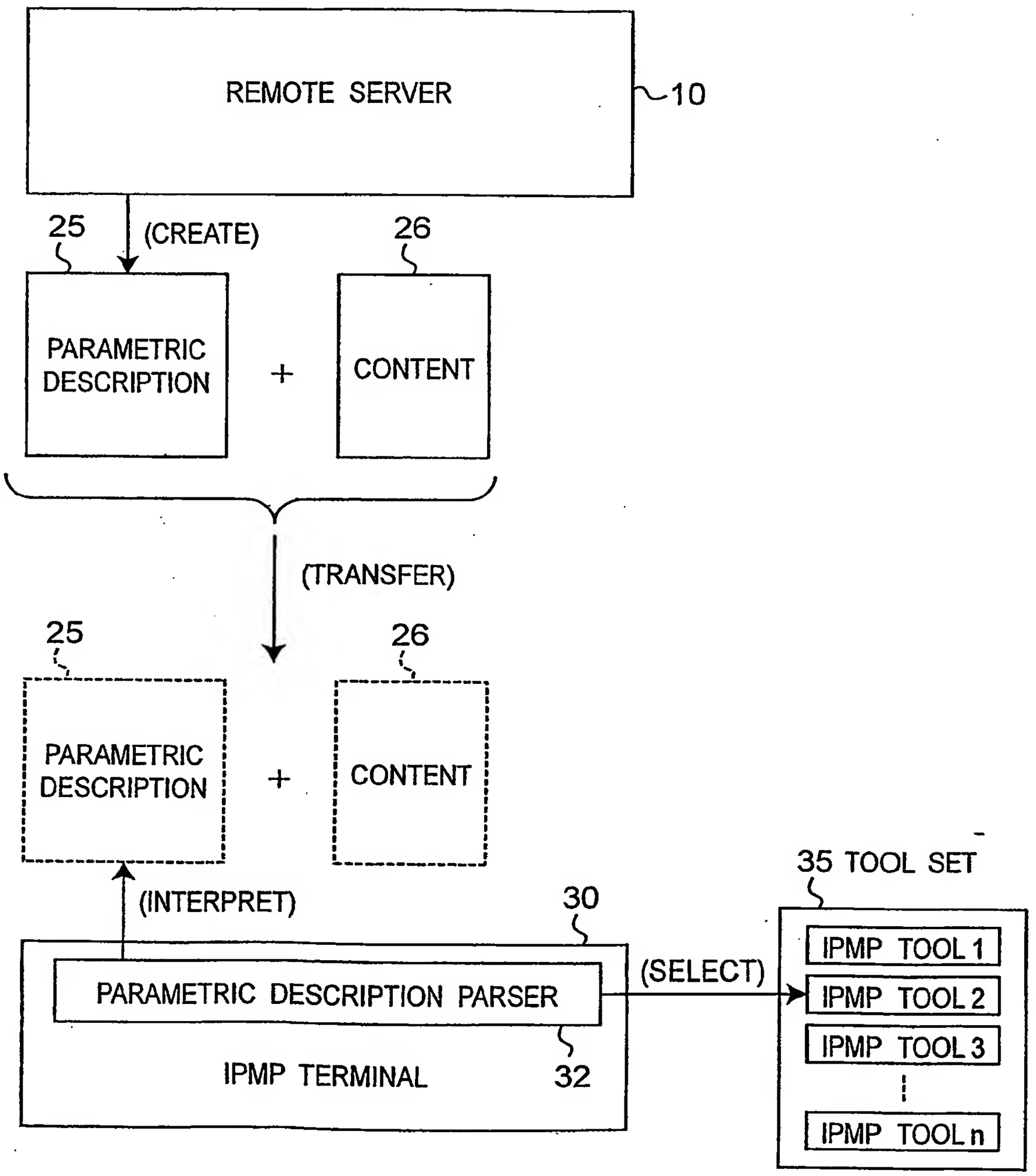
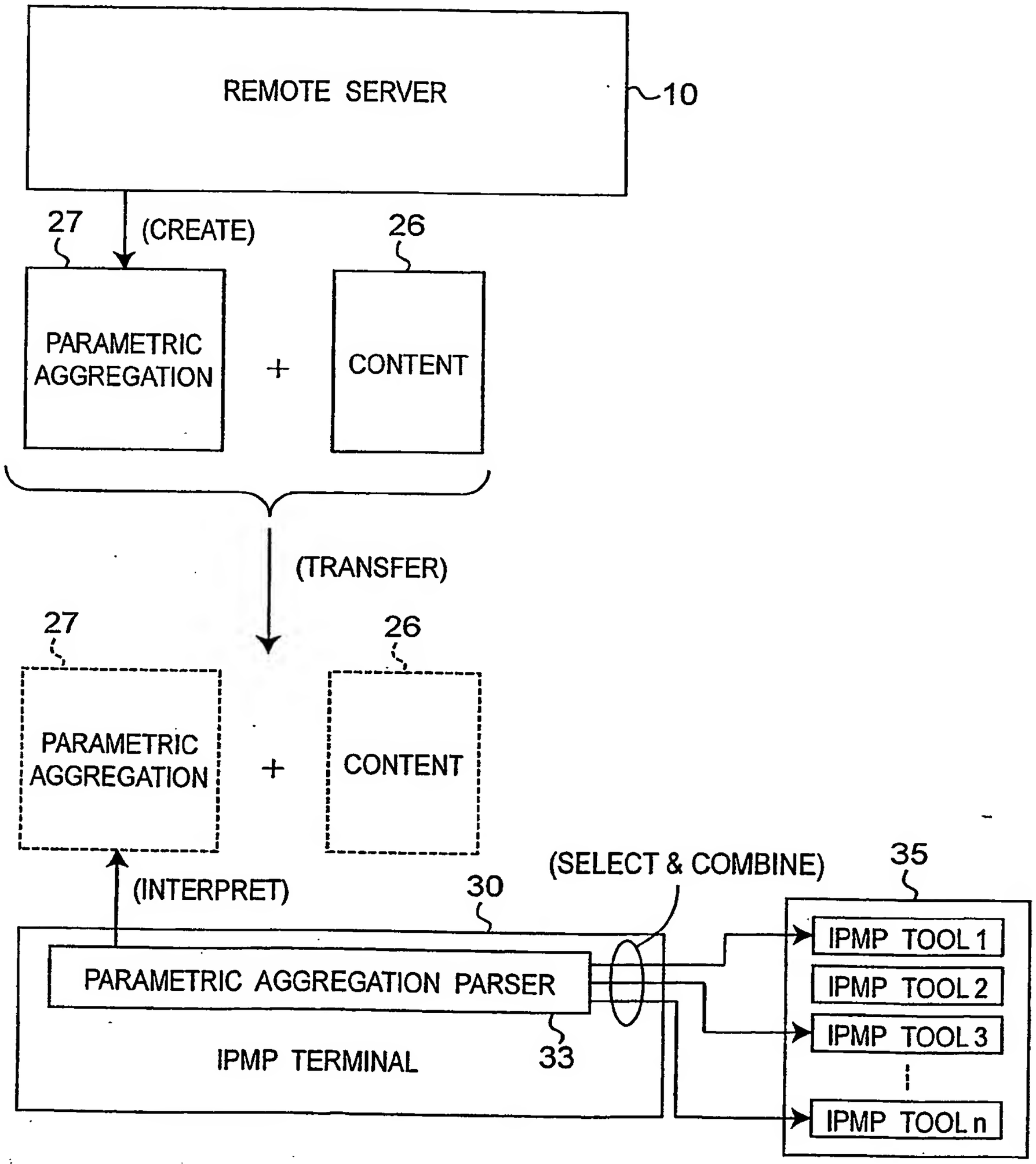

REMOTE SERVER
10
25
(CREATE)
26
PARAMETRIC DESCRIPTION
+
CONTENT
(TRANSFER)
25
26
PARAMETRIC DESCRIPTION
+
CONTENT
(INTERPRET)
30
PARAMETRIC DESCRIPTION PARSER
32
IPMP TERMINAL
(SELECT)
35 TOOL SET
IPMP TOOL 1
IPMP TOOL 2
IPMP TOOL 3
IPMP TOOL n

Fig.4


REMOTE SERVER
10
27
(CREATE)
26
PARAMETRIC AGGREGATION
+
CONTENT
(TRANSFER)
27
26
PARAMETRIC AGGREGATION
+
CONTENT
(INTERPRET)
30
(SELECT & COMBINE)
35
PARAMETRIC AGGREGATION PARSER
33
IPMP TERMINAL
IPMP TOOL 1
IPMP TOOL 2
IPMP TOOL 3
IPMP TOOL n

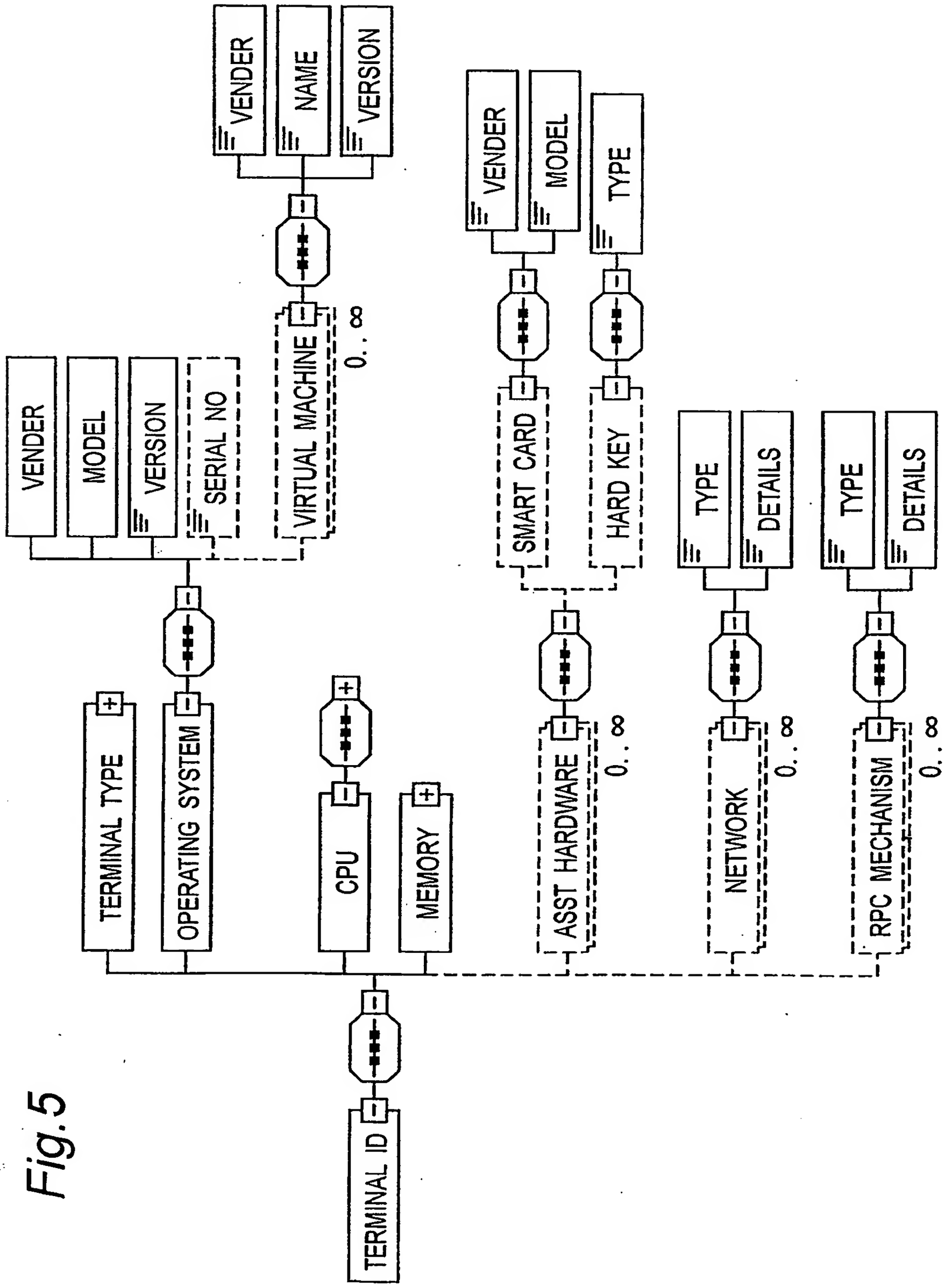

TERMINAL ID
VENDER
NAME
VERSION
VENDER
MODEL
TYPE
VENDER
MODEL
VERSION
SERIAL NO
VIRTUAL MACHINE
0..∞
SMART CARD
HARD KEY
TYPE
DETAILS
TYPE
DETAILS
TERMINAL TYPE
OPERATING SYSTEM
CPU
MEMORY
ASST HARDWARE
0..∞
NETWORK
0..∞
RPC MECHANISM
0..∞

Fig.5

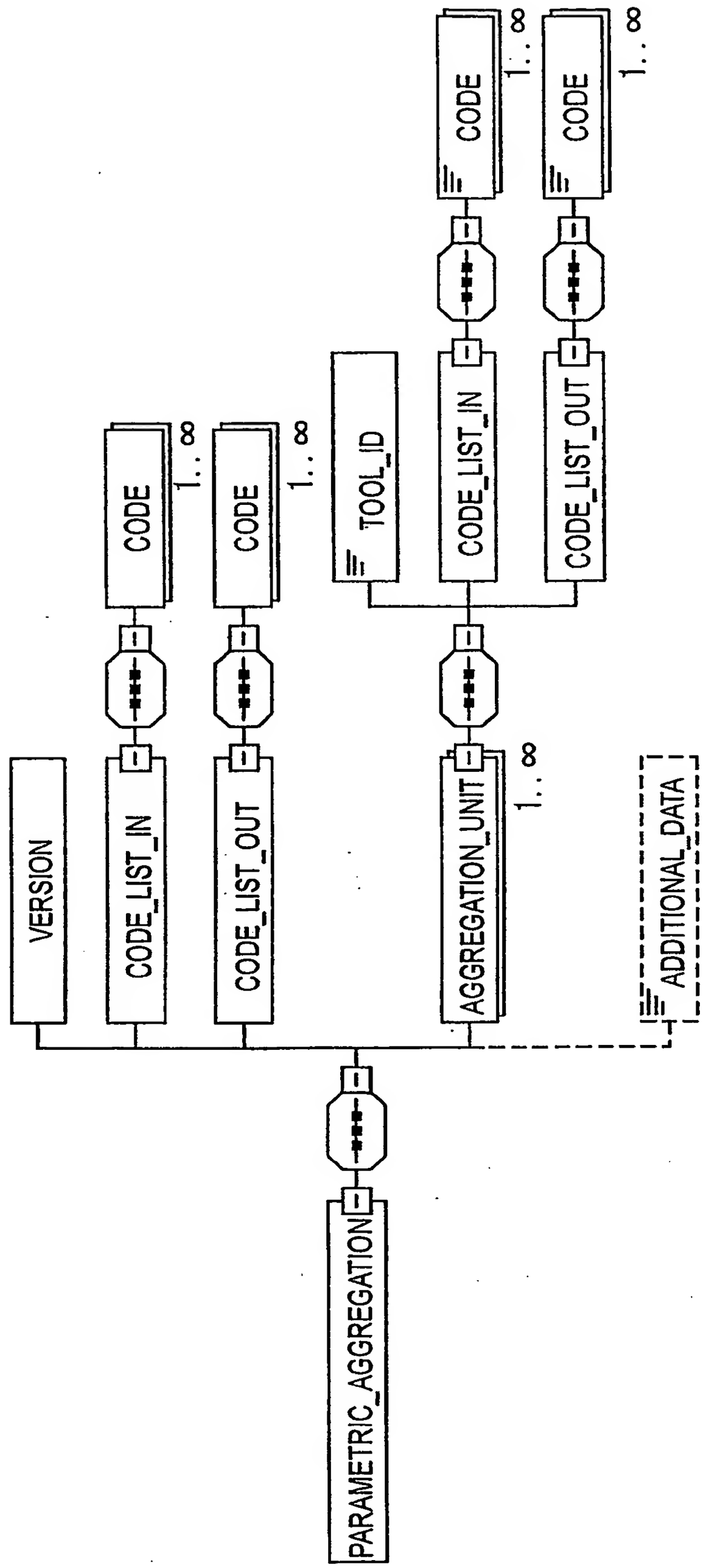

PARAMETRIC_AGGREGATION
VERSION
CODE_LIST_IN
CODE_LIST_OUT
CODE
1..8
CODE
1..8
TOOL_ID
CODE_LIST_IN
CODE_LIST_OUT
AGGREGATION_UNIT
1..8
ADDITIONAL_DATA

Fig.6

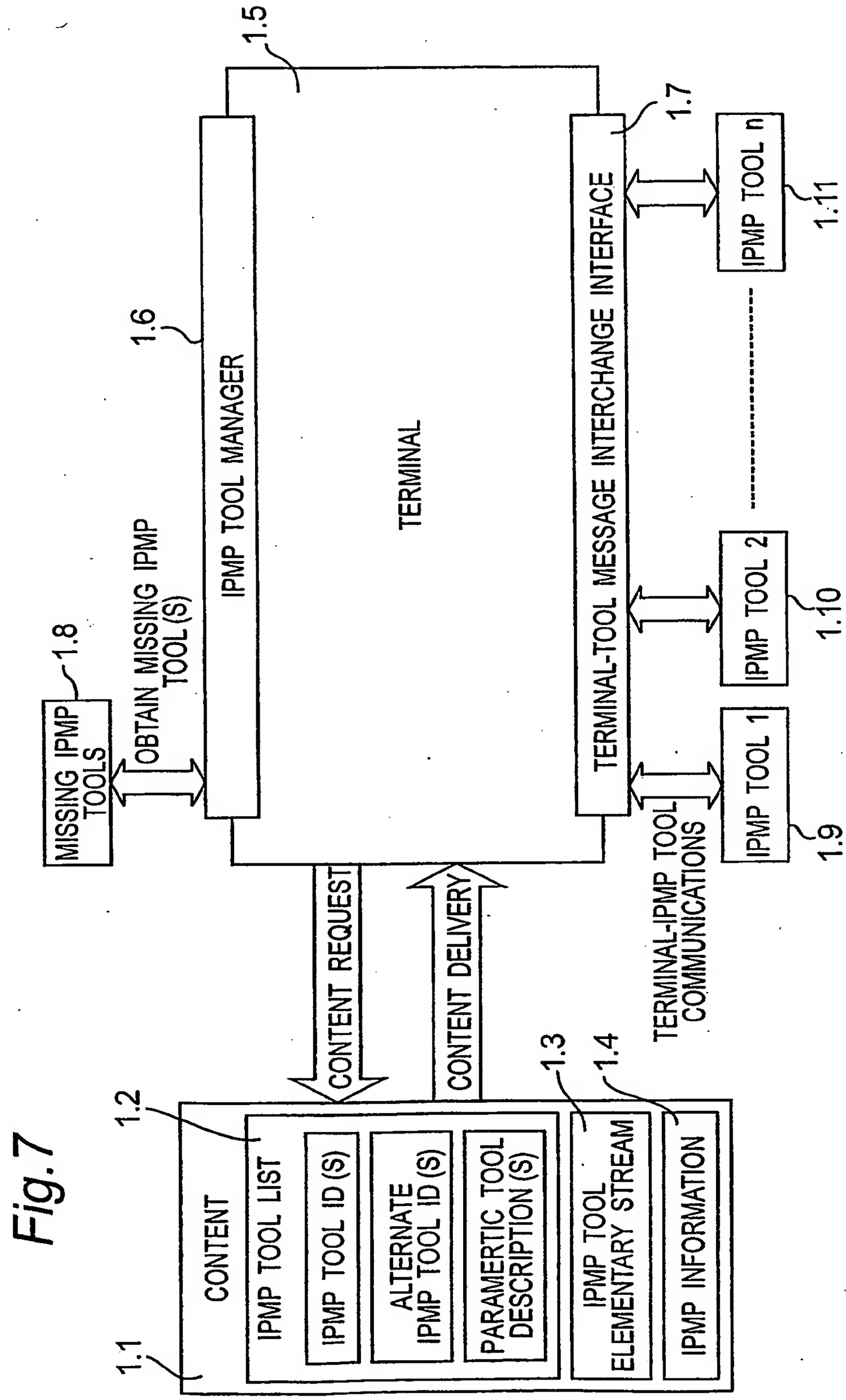

CONTENT
IPMP TOOL LIST
IPMP TOOL ID (S)
ALTERNATE IPMP TOOL ID (S)
PARAMERTIC TOOL DESCRIPTION (S)
IPMP TOOL ELEMENTARY STREAM
IPMP INFORMATION
1.1
1.2
1.3
1.4
CONTENT REQUEST
CONTENT DELIVERY
TERMINAL
IPMP TOOL MANAGER
1.6
OBTAIN MISSING IPMP TOOL(S)
MISSING IPMP TOOLS
1.8
1.5
TERMINAL-TOOL MESSAGE INTERCHANGE INTERFACE
1.7
TERMINAL-IPMP TOOL COMMUNICATIONS
IPMP TOOL 1
1.9
IPMP TOOL 2
1.10
IPMP TOOL n
1.11

Fig.7